Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Titulació:

Enginyeria Industrial (pla 2003)

Alumne (nom i cognoms):

Marcelo Germán López Pérez

Títol PFC:

Proyecto de diseño de una red inalámbrica de sensores de bajo coste

Director del PFC:

David González Diez

Convocatòria de lliurament del PFC:

2014/15 Q1

Contingut d'aquest volum: **-MANUAL DE USUARIO-**

Índice de contenido

1. El nodo sensor 4
   1.1. Hardware 4
   1.2. Construcción 5
   1.3. Programación del nodo 6
2. Configuración de las radios 8
3. Software de procesado de datos 12
   3.1. Nivel de red 12
   3.2. Nivel de nodo 13
   3.3. Nivel de dispositivo 15
   3.4. Archivos 17
4. Referencia de interés 19
Anexos 1
   Formato de datos 1
   Comandos de actuación 6

Índice de figuras

Figura 1. XBee Explorer Regulated. Fuente: SparkFun. 4
Figura 2. Módulos utilizados para construcción de un nodo sensor. 6
Figura 3. Aspecto del nodo sensor montado. 6
Figura 4. Ventana de X-CTU en el modo de configuración 8
Figura 5. Mensaje solicitando resetear el módulo XBee 9
Figura 6. Ventana principal de la aplicación de procesado de datos. 12
Figura 7. Ventana de edición para el menú de red 13
Figura 8. Ejemplo de etiquetas de los tres tipos de nodo de la red. 13
Figura 9. Ventana de la aplicación en el menú del nodo. 14
Figura 10. Ventana de edición para el menú del nodo. 14
Figura 11. Ejemplo de etiquetas para los dispositivos de un nodo. 14
Figura 12. Ventana de la aplicación en el menú del dispositivo 15
Figura 13. Ventana de la aplicación para un actuador. 16

Índice de tablas

Tabla 1. Ejemplo de configuración de los parámetros de un módulo Xbee. 10
Tabla 2. Estructura utilizada para la transmisión de mensajes..... 1
Tabla 3. Codificación propuesta de los data type. 1
Tabla 4. Estructura del formato para la transmisión de datos..... 3
Tabla 5. Código de la función ftohex. 3
Tabla 6. Función de control del potenciómetro. 4
Tabla 7. Función de control del sensor DHT-111..... 5
Tabla 8. Estructura del mensaje de actuación para un led RGB. 7
Tabla 9Funciones utilizadas para comandar el led RGB..... 7

El presente manual se ha desarrollado con el fin de facilitar la tarea del usuario a la hora de programar y operar el sistema desarrollado. Aquí se describen todos los pasos necesarios para instalar un nodo sensor, la configuración de la red de comunicaciones multi-hop y del programa de procesado de datos. También se dan algunos consejos para facilitar el uso y ayudar a entender el funcionamiento del sistema.

1. El nodo sensor

El nodo sensor es el encargado de realizar las mediciones desde los sensores, procesar la información, enviar los datos al nodo central de la red y realizar alguna acción correctora, si procede.

1.1. Hardware

El nodo sensor utilizado en este sistema tiene una construcción bastante sencilla y modular. Consiste en una placa Arduino, encargado de las funciones de micro controlador, y un módulo XBee ZB conectados mediante el necesario adaptador.

La placa Arduino utilizada para desarrollar este proyecto fue el modelo UNO, aunque cualquier otro modelo que utilice un MCU de la misma familia que el AT Mega 328 P podría funcionar igualmente.

Las radios utilizadas en los nodos sensores deberán ser XBee de la serie 2 bajo protocolo ZigBee. Para funcionar correctamente las radios deberán estar configuradas con el firmware en modo API. Puede utilizarse cualquiera de las diferentes versiones de antena disponibles, pues son intercambiables.

En cuanto al adaptador, se recomienda utilizar el que mejor se adapte a las necesidades del usuario. Existen diferentes modelos, como la Wireless Shield desarrollada por el Team Arduino, o el XBee explorer desarrolado por SpakFun.

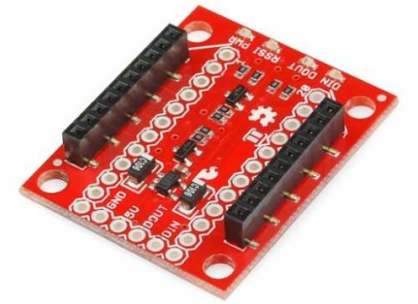

Figura 1. XBee Explorer Regulated. Fuente: SparkFun.

El usuario debe recordar que el adaptador es un componente necesario no sólo porque la separación de los pines del módulo XBee es menor a la utilizada normalmente en otros componentes como Arduino o protoboards, sino porque operan con niveles lógicos diferentes. El Arduino uno opera con lógica de 5V, mientras los módulos XBee operan con lógica de 3,3 V. Esto dificulta la conexión directa de estos dispositivos y su buen funcionamiento, a menos que se utilice un adaptador lógico de niveles para solucionar el problema. Por esta razón la gran mayoría de adaptadores comerciales para XBee incorporan adaptadores lógicos en sus circuitos.

1.2. Construcción

La construcción del nodo es bastante sencilla. Simplemente hay que conectar los módulos para asegurar la comunicación entre los UART del Arduino y el XBee. Esto se consigue conectando el pin 2 (DOUT) del módulo XBee al pin 0 de Arduino (RX), y el pin 3 (DIN) del módulo XBee al pin 1 de Arduino (TX). Si el usuario trabaja con una Shield del Team Arduino deberá colocar los Jumpers en la posición XBee; o si utiliza las nuevas versiones, deberá colocar el conmutador en la posición MICRO.

También se debe suministrar energía a la radio conectando el pin 1 del módulo XBee (VCC) al pin 3,3 V de la placa Arduino, y el pin 10 (GND) al respectivo pin del micro.

Una última conexión que se debe realizar para emplear el modo SLEEP o de ahorro de energía es entre el pin 13 del módulo XBee (ON/OFF) y el pin digital 2 de la placa Arduino. Esta conexión se debe realizar utilizando una puerta inversora como la sn74ls04 de Texas Instruments. Este tema se explicará en detalle en el apartado SLEEP.

A modo de ejemplo se puede ver en la siguiente imagen los componentes de un nodo sensor y su aspecto una vez montado.

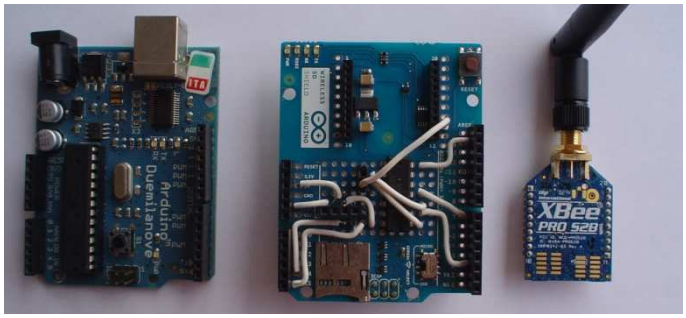

Figura 2. Módulos utilizados para construcción de un nodo sensor.

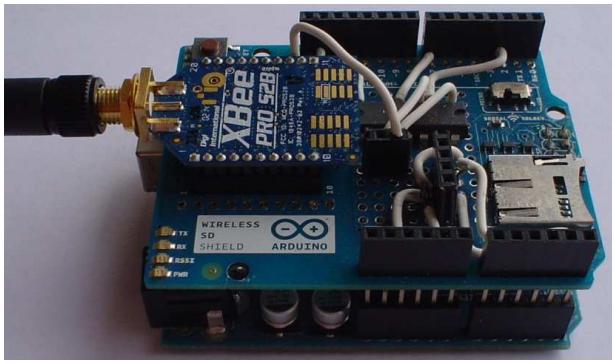

Figura 3. Aspecto del nodo sensor montado.

1.3. Programación del nodo

Junto a la documentación del proyecto se suministra el archivo de Arduino con el código base a partir del cual el usuario puede desarrollar su aplicación. El usuario deberá desarrollar las funciones para realizar las mediciones de los dispositivos de su aplicación siguiendo el formato de datos de 5 bytes explicado en el anexo, donde también se muestran ejemplos.

Por otro lado, el usuario también puede programar funciones para realizar acciones comandadas a partir de mensajes recibidos siguiendo el formato de datos para la actuación explicado en el anexo, donde también se muestran ejemplos.

Todo dispositivo conectado al nodo, ya sea sensor o actuador, deberá de tener asignado un número de identificación o ID único para el nodo. Éste será un número positivo de 8 bits, un byte, que se utilizará para identificar los mensajes que se envíen o las instrucciones que se reciban, de acuerdo al formato de datos establecido.

Uno de los principales parámetro que el usuario deberá definir en la programación del nodo sensor es la dirección del nodo coordinador de la red.

Este parámetro se define escribiendo la dirección de 64 bits de la radio Xbee coordinador (que puede verse en el reverso del módulo), más la dirección de 16 bits de este nodo (0x00, 0x00).

Otra de las características que el usuario deberá definir durante la programación del nodo es si éste realizará ciclos de SLEEP o no. Si el usuario opta por mantener el nodo encendido en todo momento deberá definir la frecuencia de muestreo del nodo en la variable `frecuencia`. Esta variable define la frecuencia de muestreo en milisegundos, pudiendo adoptar valores entre 0 y 65535. Por el contrario, si el usuario opta por configurar el modo SLEEP, deberá asegurarse de realizar las conexiones indicadas en el apartado de construcción y proceder a la configuración de la radio. En cualquier caso, el usuario deberá seleccionar en la función `loop()` la condición del `if()` dependiendo del modo escogido, comentando la otra .

2.Configuración de las radios

Las radios se configurarán con la aplicación desarrollada por DIGI para tal fin: X-CTU. Se recomienda utilizar la versión 6.1.1 o superior por ser más cómoda de trabajar.

Para configurar una radio XBee utilizando la placa Arduno como Adaptador, hay que realizar las conexiones de datos de tal manera que el UART del XBee esté conectado con el puerto USB de la placa. Para ello, las conexiones con la placa deben intercambiarse, conectando DOUT a TX y DIN a RX. Si se trabaja con una Shield simplemente coloque los jumpers o el conmutador en la posición USB. Para evitar interferencias es necesario extraer el micro controlador de la placa.

Una vez conectada la placa al puerto USB de su ordenador, siga las instrucciones de X-CTU para agregar el nuevo módulo. En la pestaña de configuración podrá ver y editar los parámetros de la radio. Recuerde que muchos de los parámetros se escriben en formato hexadecimal.

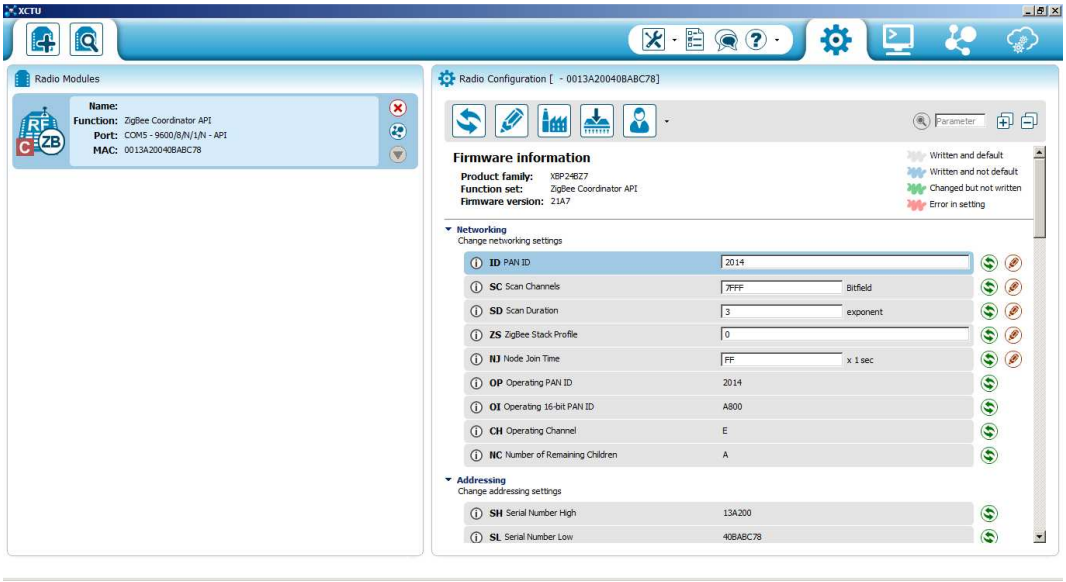

Figura 4. Ventana de X-CTU en el modo de configuración

El primer paso para configurar un módulo XBee para la red es cargar el firmware en el dispositivo. Para ello seleccione en la pestaña de configuración. A continuación se abrirá la ventana de actualización del firmware donde deberá seleccionar la familia a la que pertenece su XBee (puede encontrarla en la parte inferior del módulo); al función del nodo en la red y la versión del firmware (seleccione siempre versiones por encima de 2170). En cuanto a la función, deberá seleccionarla en función del rol que seguirá la radio en la red ZigBee:

coordinador, router o end device, y dependiendo de las características de su nodo sensor. Destacar que en una red ZigBee sólo hay un coordinador; los router tienen la capacidad de ejecutar ciclos de SLEEP y de pasar mensajes ente radios en una red multi-hop; y los end device siempre funcionan con el modo SLEEP habilitado y no pueden pasar mensajes como los routers.

Si durante la carga del firmware o en cualquier otro momento aparece un mensaje como el de la siguiente imagen, deberá proceder a resetear el módulo Xbee. Si utiliza un adaptador específico puede que disponga de un switch para ello. En otros casos como en las shields, deberá proceder puenteando el pin 5 del Xbee (RESET) a GND. No se recomienda estraer el Xbee de su zócalo para resetearlo ya que esto puede provocar problemas no sólo en el módulo sino en su ordenador. Si el mensaje persiste revise las conexiones, puede tratarse de un problema de montaje.

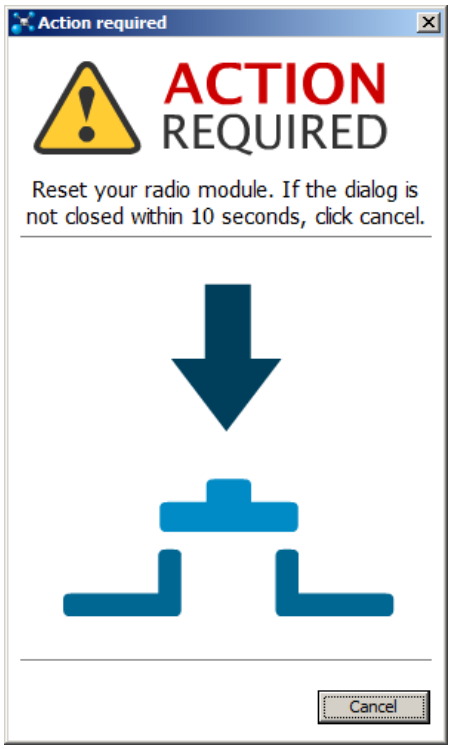
Action required
ACTION
REQUIRED
Reset your radio module. If the dialog is not closed within 10 seconds, click cancel.
Cancel

Figura 5. Mensaje solicitando resetear el módulo XBee.

Una vez cargado el firmware se procederá a configurar los parámetros de la radio. Lo primero que se debe definir el la PAN ID o dirección de la red en el apartado Networking. Luego se debe definir la dirección de destino de la radio en el apartado Adressing, indicando la parte alta y baja de la dirección de 64 bits de la radio a la que se conectará el nodo. Este parámetro es particularmente

importante para los end device, ya que requieren de un nodo padre para poder conectarse a la red, el cual debe ser un router o el coordinador.

En el apartado de Serial Interfacing deberá configurar el parámetro AP a valor 1 para que la radio funcione correctamente con la programación desarrollada para las comunicaciones en modo API en el nodo sensor y en el programa de procesado de datos.

Finalmente, en el apartado Sleep Mode se configuran, si procede, los parámetros del ciclo de SLEEP que realizará el nodo. En este sistema se utilizará siempre el modo Sleep cíclico: SM=4. El tiempo que el nodo permanecerá apagado se calcula con la siguiente fórmula: SP·SN·10 (tiempo apagado por número de ciclos seguidos por 10), cuando SO=4 dando como resultado el tiempo en milisegundos. Por defecto (SO=0), el módulo sólo está apagado el tempo indicado por SP, mientras que en el modo avanzado el tiempo se multiplica por SN.

Por su parte, el tiempo que el nodo permanecerá encendido se define por el parámetro ST en milisegundos. Este tiempo será bastante corto en la mayoría de aplicaciones con modo SLEEP habilitado para ahorrar energía.

A modo de ejemplo, si un nodo end device está conectado a la red cuya ID es 0x2014, a través de un router con dirección 0x0013A200 0x40BABC80 y utiliza ciclos SLEEP permaneciendo 280 segundos apagado y 5 segundos encendido, esta sería la configuración de los parámetros:

Tabla 1. Ejemplo de configuración de los parámetros de un módulo Xbee.

Parámetro	Valor
ID	2014
DH	0013A200
DL	40BABC80
AP	2
SM	4
SN	0A
SO	04
SP	AF0
ST	1388

Adicionalmente, se debe configurar el apartado Sleep Mode del coordinador de la red con los valores de los parámetros SN y SP mayores utilizados en los nodos. Esto debe hacerse porque si el coordinador no recibe respuesta de un nodo tras el tiempo establecido por sus parámetros SN y SP, según la fórmula del Sleep mode, eliminará el nodo de la lista de dispositivos de la red. Cuando eso ocurre el nodo debe volver a realizar todo el proceso de asociación a la red antes de poder enviar sus mensajes, lo que retrasa las comunicaciones.

Una vez configurada la red, el nodo coordinador, formado por el Xbee configurado como coordinador y conectado a una placa Arduino sin MCU, irá conectado al ordenador donde se ejecutará la aplicación de procesado de dato, mientras que el resto de nodos se situarán donde así lo requiera la aplicación.

NOTA: Si al conectar el nodo sensor éste no transmite los mensajes de los sensores, es posible que se deba a que Arduino no ha procesado correctamente el mensaje de modem status que se recibe al arrancar el XBee y conectarse a la red. En ese caso puede enviar un comando AT remoto con X-CTU para resetear el XBee y que vuelva a enviar un mensaje de modem status a Arduino, o puede optar por resetear manualmente el módulo puenteando el pin 5 del Xbee a GND.

3.Software de procesado de datos

La aplicación de procesado de datos se ha desarrollado con el fin de facilitar su uso al usuario. Cuando el usuario inicia la aplicación por primera vez verá la siguiente imagen:

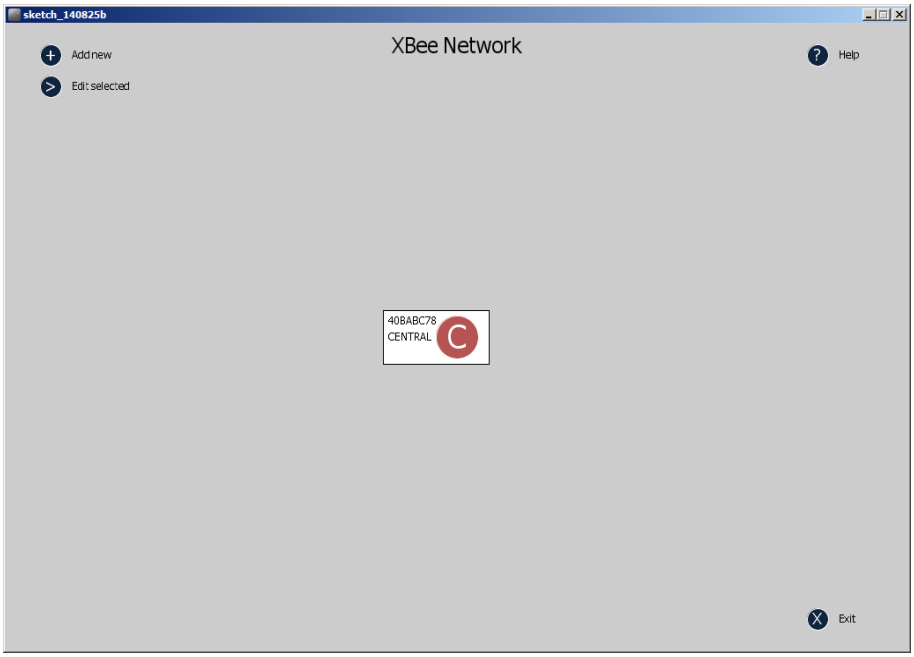

Figura 6. Ventana principal de la aplicación de procesado de datos.

3.1. Nivel de red

En la pantalla principal o menú de red aparecerán todos los nodos de los que se compone la red de sensores. Inicialmente sólo aparece el coordinador de la red, pues es el único imprescindible.

En esta ventana pueden verse otros cuatro botones: el botón de Exit, que finaliza la aplicación; el botón Help que inicial este manual; y dos botones de edición: Add new y Edit selected. Cualquiera de ellos ejecuta inicia el modo de edición que permite agregar nuevos nodos o editar existentes.

Para agregar un nodo, se indicará como ID la parte baja de la dirección de 64 del módulo XBee del nodo. El nombre que se le asigne al nodo es elección del usuario. El rol del se corresponderá con el rol del Xbee del nodo. Finalmente, se colocará la etiqueta del nodo en la posición que se desee en la pantalla mediante Place label. Presione este botón y aparecerá una etiqueta que seguirá la posición

del ratón por la ventana. Ara fijar la posición simplemente clique en la posición que desee. Puede recolocar una etiqueta de la misma forma.

En modo edición se pueden editar los parámetros de uno de los nodos: su número de ID, nombre, role y posición de la etiqueta en la ventana. También se puede eliminar un nodo, siempre que no sea el coordinador, borrando su ID y aceptando los cambios.

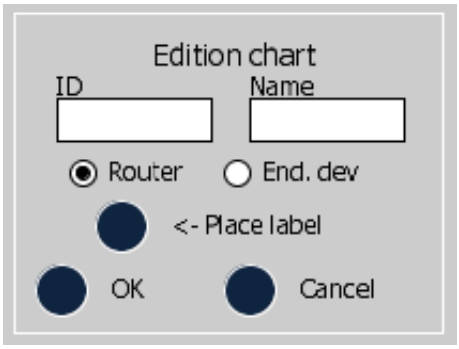

Figura 7. Ventana de edición para el menú de red

Figura 8. Ejemplo de etiquetas de los tres tipos de nodo de la red.

3.2. Nivel de nodo

Al seleccionar un nodo, presionando sobre el botón circular de su etiqueta, se pasa al menú del nodo donde pueden verse todos los dispositivos conectados al nodo.

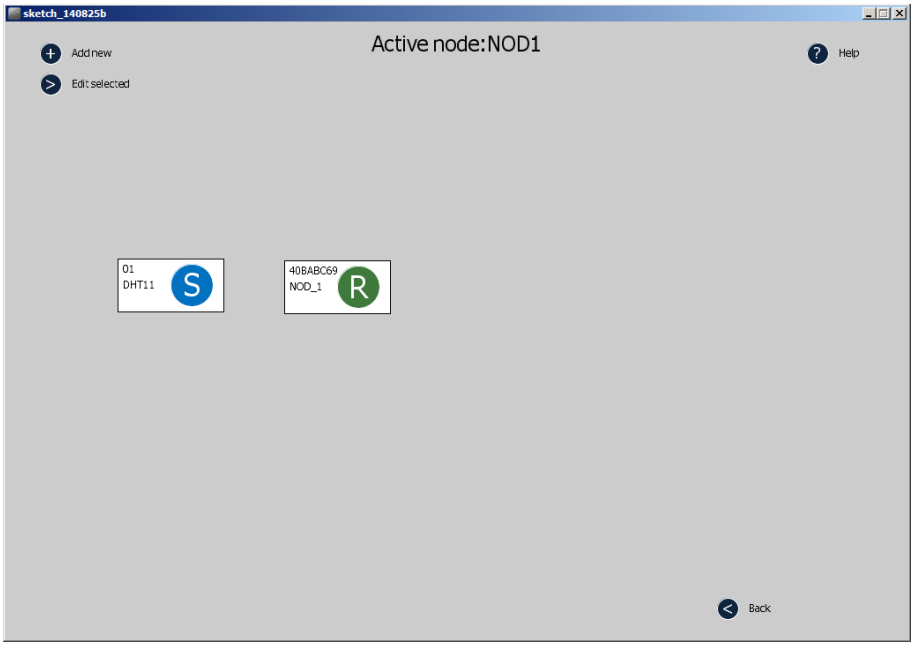

Figura 9. Ventana de la aplicación en el menú del nodo.

En este modo también se pueden agregar y editar los dispositivos de forma similar al menú de red. La diferencia en este caso es que los roles admitidos para los dispositivos son sensor y actuador.

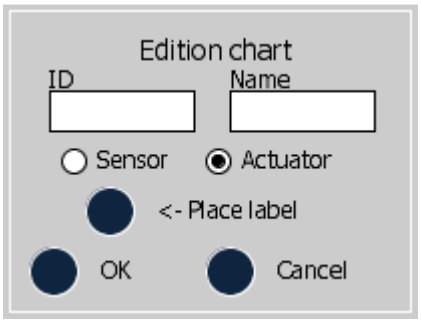

Figura 10. Ventana de edición para el menú del nodo.

Para agregar un nuevo dispositivo en este caso, la ID del dispositivo deberá coincidir con la ID asignada por el usuario al dispositivo durante la programación del nodo. De esta manera los datos se podrán procesar y recopilar correctamente.

Figura 11. Ejemplo de etiquetas para los dispositivos de un nodo.

Los parámetros de los dispositivos se pueden editar de la misma forma que en el menú de red.

3.3. Nivel de dispositivo

Al seleccionar un dispositivo, presionando también sobre el botón circular de su etiqueta, se pasa al menú de dispositivo donde pueden verse todos los parámetros configurados de los que se recibirán datos.

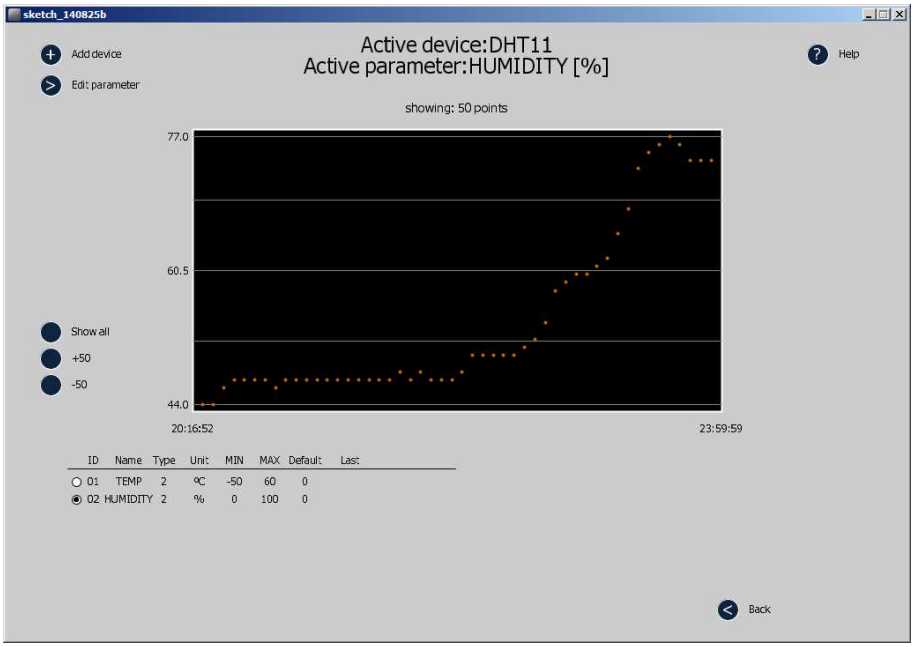

Figura 12. Ventana de la aplicación en el menú del dispositivo.

En este menú también pueden verse todas les mediciones recibidas del dispositivo en el gráfico de datos. El gráfico muestra las mediciones del parámetro actualmente seleccionado, que se pude cambiar seleccionando otro de los parámetros de la lista.

Para el control del gráfico se disponen aquí de tres botones. Show all alterna entre mostrar todas las mediciones recibidas o sólo un número limitado, el cual se puede aumentar en 50 puntos (hasta un máximo de 500) con el botón +50, o disminuir en 50 puntos (hasta un mínimo de 50) con el botón -50.

En el caso de que el dispositivo sea un actuador, el menú del dispositivo incluye el comando de actuación. Esta función permite seleccionar el nivel al que se desea colocar uno de los parámetros del dispositivo y enviar una instrucción al nodo sensor para ejecutar la acción.

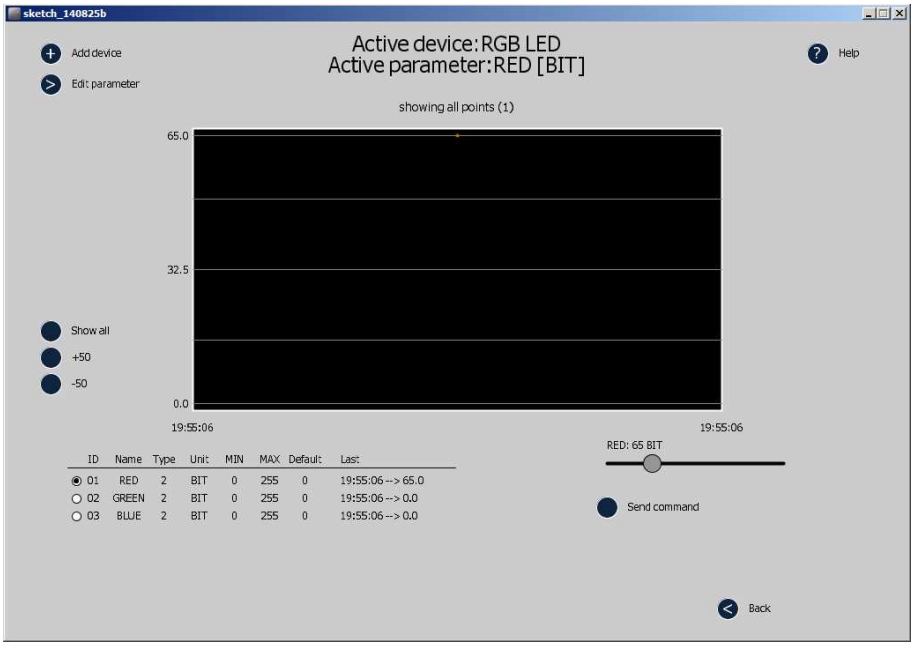

Figura 13. Ventana de la aplicación para un actuador.

En este nivel también se pueden agregar y editar las propiedades de los parámetros. En este caso, la ID del parámetro se asigna automáticamente. En el apartado de tipo, se reconocen dos clases:

- **1**: parámetro binario. Son parámetros de naturaleza booleana. Estos parámetros sólo pueden adoptar dos valores, o y 1, que se asignarán a cada posición en la programación del nodo.
- **2**: parámetro variable. Son aquellos parámetros que pueden variar en el rango establecido por el máximo y el mínimo. Las variables pueden ser de naturales integer o float.

El usuario deberá asignar también una unidad de medida al parámetro y se puede asignar una posición por defecto del parámetro, que se puede tener en cuenta a la hora de realizar actuaciones.

A diferencia de los otros niveles de la aplicación, en este nivel no se puede eliminar un parámetro directamente desde el cuadro de edición. Como medida de seguridad, la eliminación de los parámetros se debe realizar manualmento modificando el archivo del dispositivo.

3.4. Archivos

Esta aplicación funciona con cuatro tipos de archivos:

- `Network.csv`: es el archivo principal en el que se guardan todos los nodos de la red y sus características. Ejemplo:

```
ID,name,X,Y,role
40BABC78,CENTRAL,400,300,0
40BABC69,NOD1,568,300,1
40BABC33,NOD2,234,300,1
```

- `Device.csv`: es el archivo en el que se guardan todos los dispositivos conectados al nodo y sus características. Se encuentra dentro de la carpeta con la ID del nodo correspondiente, dentro de la carpeta nodes. Ejemplo:

```
ID,name,X,Y,role
40BABC69,NOD_1,400,300,1
01,DHT11,211,298,3
```

- Parameters: es el archivo en el que se guardan todos los parámetros del dispositivo y sus carñacteristicas. Se encuentra dentro de la carpeta con la ID del nodo correspondiente, dentro de la carpeta nodes, y el archivo se identifica por la ID del dispositivo (ejemplo: `01.csv`, para el dispositivo con ID 01). Ejemplo:

```
ID,name,type,unit,min,max,default,last
01,TEMP,2,ºC,-50,60,0,
02,HUMIDITY,2,%,0,100,0,
```

- Data: es el archivo en el que se guardan todas las mediciones recibidas desde los nodos sensores para un dispositivo. Este archivo se genera automáticamente para cada dispositivo cada vez que se inicia la aplicación. Se encuentra dentro de la carpeta con la ID del nodo correspondiente, dentro de una carpeta con la fecha del día, y el archivo se identifica por la ID del dispositivo (ejemplo: `01.csv`, para el dispositivo con ID 01). Ejemplo:

```
time,01,02
del,0,0
08:29:04,23.0,52.0
08:33:42,22.0,54.0
08:38:20,22.0,55.0
08:42:58,22.0,56.0
```

Dado que cada vez que inicia el programa se crean los archivos de datos, la lista de parámetros del dispositivo debe estar correctamente actualizada, ya que los datos se procesan y guardan en el archivo en función de su posición en el mensaje, tal como se especifica en el formato de datos.

Eliminar algún parámetro de la lista implica reasignar las ID de los que estén por debajo de este en el archivo de parámetros. Cambiar la posición de los parámetros en la lista también puede traer problemas al procesar los datos. Si se reciben más mediciones que parámetros se ha configurado, el programa lo considera como un error al no poder guardarlos correctamente en el archivo de datos. Puede incluso llegar a cerrar el puerto del coordinador y cortar las comunicaciones, dejando de recibir cualquier dato, o directamente colgarse y dejar de funcionar.

Por esta razón, antes de comenzar a enviar datos desde un nodo sensor, se debe obligatoriamente configurar el nodo, el dispositivo y los parámetros.

Otra característica de la aplicación es que está diseñada para funcionar de forma ininterrumpida. Si por cualquier motivo la aplicación debe cerrarse, al iniciarse de nuevo durante el transcurso del mismo día, el programa vuelve a generar los archivos de datos, sobrescribiendo con archivos vacios los existentes. Por ello se recomienda pasar a un lugar seguro todos los datos recogidos antes de volver a ejecutar la aplicación de procesado de datos.

4.Referencia de interés

[1] R. Faludi, Building Wireless Sensor Networks, O'Reilly Media, 2010.

[2] Digi International Inc, «XBee ZB User Manual,» 2012. [En línea]. Available: http://ftp1.digi.com/support/documentation/90000976_T.pdf. [Último acceso: junio 2014].

[3] Processing, «References,» 2014. [En línea]. Available: http://www.processing.org/reference/. [Último acceso: junio 2014].

[4] Arduino, «Languaje References,» 2014. [En línea]. Available: http://arduino.cc/en/Reference/HomePage. [Último acceso: junio 2014].

Anexos

Formato de datos

Dentro del marco API de transmisión de mensajes, los datos de las mediciones, que forman la RF data, se configuran siguiendo la siguiente estructura.

Tabla 2. Estructura utilizada para la transmisión de mensajes.

Data type	Device	A-S data	
Byte 1	Byte 2	Byte 4…	…Byte n
Single Byte	Single byte	Data bytes	

En esta estructura el primer byte se refiere al tipo de datos que se recibirán, como mediciones provenientes de sensores, posición o estado de actuadores o valores de parámetros. Por ello se propone la siguiente codificación, aunque en evoluciones posteriores del sistema se podrán agregar opciones adicionales.

Tabla 3. Codificación propuesta de los data type.

Data type	Descripción
0x01	Datos de medidas de sensores.
0x02	Estado y/o posición de los actuadores.
0x03	Valor actual de un parámetro.

El siguiente Byte identifica el dispositivo (sensor, actuador u otro) del que provienen los datos. De esta manera, cada dispositivo conectado al nodo sensor deberá identificarse con un byte único para transmitir datos cuando se programe cada aplicación. Este byte servirá, también, para identificar los datos en el nodo central y almacenarlos correctamente.

Finalmente aparecen los propios datos a transmitir. Como se comentó en apartados anteriores, los datos sólo incluirán los valores numéricos, mientras que otras características como unidades se asociarán a la hora de almacenar los datos en el nodo central, a partir del identificador del dispositivo.

En Arduino los números enteros o `int` ocupan 16 bits; las variables de tipo `long`, 32 bits; y los `floats`, también 32 bits. Esto supone una limitación para el tamaño de los números con el que se puede trabajar y es un aspecto muy importante a la hora de transmitir los datos a través de la red. Teniendo en cuenta que las radios XBee en modo API requieren comunicarse a través de bytes, se plantearon el llamado sistema de los 54 bytes para descomponer las variables.

En esta forma se plantea dividir los números a transmitir en bytes mediante operaciones `Bitwise`. De esta forma, cualquier número entero se podría transmitir utilizando sólo dos bytes, dividiéndolo en su parte alta y baja. Las variables de tipo `long` se podrían tratar de forma similar pero dividiéndola en 4 bytes.

Ejemplo:

- -1024 →0xFC 0x00 (1111 1100 0000 0000)
- 40→ 0x00 0x28 (0000 0000 0010 1000)

El problema de esta forma es que aunque Arduino trabaja con enteros de 16 bits, los ordenadores para el almacenamiento de datos son sistemas de 32 bits. Esto hace difícil reconocer y reconstruir números negativos enviando sólo dos bytes de información. Por esta razón se incluye un byte adicional de signo. De esta forma, cuando este byte tenga valor 0x01 será indicativo de un número negativo, mientras el número en sí se enviará en valor absoluto

- -1024 →0x01 0xFC 0x00 (0000 0001 0000 0100 0000 0000)
- 40→ 0x00 0x28 (0000 0000 0000 0000 0010 1000)

Las variables de tipo `long` requerirán entonces cinco bytes para ser transmitidas.

Por su parte las variables de tipo float son un poco más complejas de manipular. Dado que no se puede tener acceso para partir los 32 bits de este tipo de variables en 4 bytes, se optó por una solución alternativa. Esta solución consiste en separar este tipo de variables en dos enteros, uno para su parte entera y otro para la decimal y transmitirlos como tales. Como resultado, todo float se puede trasmitir utilizando 5 bytes: uno para el signo, dos para la parte entera y otros dos para la parte decimal.

Ejemplo:

- -27.36 → 0x01 0x00 0x1b 0x00 0x24

Aunque debido a características propias de Arduino la forma en que transmitirá el número es:

- -27.36 → 0x01 0x00 0x1b 0x8C 0xA0

La parte decimal se transforma en un entero de 16 bits lo más grande posible. En este caso el equivalente entero de la parte decimal del número es:

- 0x8C 0xA0 → 36000

Teniendo en cuenta las posibles variables con las que se podrá trabajar, se planteó un marco único de cinco bytes para transmitir cualquier número, independientemente del tipo de variable del que se trate.

Tabla 4. Estructura del formato para la transmisión de datos.

Signo	0x00
Bytes superiores o parte entera	0x00
	0x00
Bytes inferiores o parte decimal	0x00
	0x00

Un número entero se transmitirá utilizando los dos bytes superiores, mientras que los bytes inferiores se utilizarán al transmitir variables `float` o `long`. De esta mera cada número siempre ocupará lo mismo y es más fácil delimitarlo y procesarlo para recuperarlo en destino mediante operaciones nversas a las aplicadas para partirlo.

El usuario deberá programar las funciones de sus dispositivos para que funcionen de acuerdo a esta estructura, ya que es la misma que se descodifica en el programa de procesado de datos. Para simplificar la tarea se puede utilizar la función auxiliar que se incluye en el programa base `ftohex()`. Dada una variable de tipo float, esta función devuelve un array de cinco bytes con los datos formateados.

Tabla 5. Código de la función ftohex.

```
void ftohex (float number, uint8_t result [5]){
  uint16_t aux;
  if(number<0){
    result[0]=0x01;
    number=abs(number);//eliminar el signo
  }
  else result[0]=0x00;
```

```
  aux=number;//extraemos parte entera
  result[1]=highByte(aux);
  result[2]=lowByte(aux);
  //extraer la parte decimal
  number=number-aux;
  //A será inferior al número máximo para 16 bits 2^16=65536
  //tambien el decimal debe ser mayor a 0, sino se quedaria
en el bucle tener en cuenta los bits!!!
  while(number<6553.6 && number>0)number=number*10;
  aux=number;
  result[3]=highByte(aux);
  result[4]=lowByte(aux);
 }
```

Ejemplos:

- Potenciómetro:

Para operar un potenciómetro como sensor se creó una función que ejecutaba la lectura del pin al que estaba conectado el componente, creaba el marco de cinco bytes y llamaba a la función para enviar el mensaje.

Tabla 6. Función de control del potenciómetro.

```
uint8_t sensor[7];

void pot_messurement(){
  int pot=analogRead(0);
  //ID del dispositivo
  sensor[0]=0x01;
  //tipo de mensaje
  sensor[1]=0x01;
  //signo
  sensor[2]=0;
  //parte entera
  sensor[3]=highByte(pot);
  sensor[4]=lowByte(pot);
  //parte decimal
  sensor[5]=0x00;
  sensor[6]=0x00;

tx_request(sensor,sizeof(sensor));
}
```

- DHT-11

Se trata de un sensor que mide temperatura y humedad y devuelve dos variables de tipo float. La función programada para el dispositivo, solicita las mediciones al sensor, llama a la función ftohex para cada parámetro, compone el mensaje y llama a la función para enviarlo.

Tabla 7. Función de control del sensor DHT-111.

```
uint8_t value[5];

void DHT_messurement(){
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  uint8_t THM[12];
  //medida
  THM[0]=0x01;
  //dispositivo
  THM[1]=0x01;
  ftohex(t, value);
  for(uint8_t i=0; i<sizeof(value); i++)THM[i+2]=value[i];
  ftohex(h, value);
  for(uint8_t i=0; i<sizeof(value); i++)THM[i+7]=value[i];

  tx_request(THM, sizeof(THM));
}
```

Comandos de actuación

Un inconveniente del programa desarrollado es que sólo se pueden ejecutar comandos de actuación un parámetro a la vez. Por otro lado, y como se explicó, los mensajes de datos de un dispositivo se envían todos juntos desde Arduino, diferenciándose unos de otros por su posición en el mensaje. El resultado es que las aplicaciones desarrolladas, tanto en Processing como en Arduino, no tratan los parámetros de forma individualizada sino como componentes de un dispositivo.

En consecuencia fue necesario desarrollar una estructura que permitiera determinar qué parámetro de un actuador se desea modificar, de acuerdo al comando enviado, mientras que el resto deberá ser ignorado.

La solución fue utilizar el primer byte de un dato como señalador. De acuerdo al formato de datos planteado de cinco bytes, el primero corresponde al signo del número, siendo este 0x00 cuando el número es positivo y 0x01 cuando es negativo. Ahora se plantea que este byte tome el valor 0xFF cuando el parámetro deba ser ignorado al transmitir un comando de actuación. De esta forma, cuando la aplicación de Arduino en el nodo sensor lea este byte entiende que no debe modificar el parámetro del actuador y que debe saltar al siguiente parámetro.

Adicionalmente esta estructura permite que los parámetros no modificados o comandados ocupen sólo un byte (0xFF) en el mensaje, mientras que el parámetro a comandar ocupará los cinco bytes, con el primero señalando el signo correspondiente.

Esta es la estructura que se programó en Processing y con la que se recibirán los mensajes en Arduino. Las funciones específicas para comandar los actuadores que se conecten al nodo sensor deberán programarse teniendo en cuenta esta estructura y adaptándose a cada caso.

A modo de ejemplo se plantea el siguiente caso:

Ejemplo:

- LED RGB

El actuador a en cuestión es un diodo led RGB con tres parámetros comandables, uno por cada color. Suponiendo que sea el segundo de los tres parámetros el que se comanda, la estructura del mensaje a enviar al nodo sensor es la siguiente:

Tabla 8. Estructura del mensaje de actuación para un led RGB.

Data type		0x02
Device		0x05
A/S data (comando de actuación)	1º param. (ignorar)	0xFF
	2º param. (actuar)	0x00
		0x00
		0x1B
		0x00
		0x00
	3º param. (ignorar)	0xFF

Cuando Arduino recibe este mensaje identificando un comando de actuación (Data type), recompone el comando y ejecuta la función correspondiente para el dispositivo señalado (0x05). Para ello se desarrolló un código que descifrara el mensaje cuando este va dirigido al dispositivo señalado.

Tabla 9Funciones utilizadas para comandar el led RGB.

```
uint8_t bright[3]={0,0,0};

void light (uint8_t R, uint8_t G, uint8_t B){
  R=R^0xFF;
  G=G^0xFF;
  B=B^0xFF;

  analogWrite(RED, R);
  analogWrite(GREEN, G);
  analogWrite(BLUE, B);
}

void LEDstate(){
  uint8_t led[17];
  led[0]=0x01;
  led[1]=actuatorLED;
  uint8_t pos=2;
  ftohex(float(bright[0]), value);
  for(uint8_t i=0; i<sizeof(value); i++)led[pos+i]=value[i];
  pos+=5;
  ftohex(float(bright[1]), value);
  for(uint8_t i=0; i<sizeof(value); i++)led[pos+i]=value[i];
  pos+=5;
  ftohex(float(bright[2]), value);
  for(uint8_t i=0; i<sizeof(value); i++)led[pos+i]=value[i];
  tx_request(led, sizeof(led));
  }
  .
```

```
  .
  .
void rx_packet(int packet_size){
  uint8_t payload[packet_size-12];
  uint8_t check=0x90;
  for(uint8_t i=0;i<11;i++)check+=Serial.read();
  for(uint8_t i=0;i<sizeof(payload);i++){
    payload[i]=Serial.read();
    check+=payload[i];
  }
  check+=Serial.read();//sumar checksum

  if(check==0xFF){
    switch (payload[0]){
      case 0x01:
                  break;
      case 0x02:  if (payload[1]==0x05){
                      uint8_t pos=2;
                      for(uint8_t j=0; j<3; j++){
                        if(payload[pos]==0xFF)pos++;
                        else{
                          bright[j]=payload[pos+2];
                          pos+=5;
                          }
                      }
                      light(bright[0], bright[1], bright[2]);
                      LEDstate();
                  }
                  break;
      case 0x03:
                  break;
      default: break;
    }
  }
}
.
.
.
```

Como puede verse, dentro de la función rx_packet se descifra el mensaje recibido. Cuando el mensaje es de tipo actuación y va dirigido al led RGB (payload[0]=0x02, payload[1]=0x05), se leen los valores para los parámetros saltando al siguiente parámetro si se recibe el byte 0xFF en primer lugar. Una vez establecidos los valores de los parámetros, se ejecuta la función light(), que aplica los cambios al led. Luego se ejecuta la función LEDstate() que devuelve el valor al que se encuentran todos los parámetros del dispositivo tras la acutación. Esta función es muy importante ya que de lo contrario el no tendría confirmación ni registro fiable de la actuación.